Cover Page

ÖREBRO UNIVERSITET

INSTITUTIONEN FÖR
NATURVETENSKAP OCH TEKNIK

Kursens namn	**Objektorienterad programmering – DT114G**
Examinationsmomentets namn/provkod	**A001**
Datum	2022-06-02
Tid	Kl. 08:15 – 13:15

Tillåtna hjälpmedel	
Instruktion	
Viktigt att tänka på	
Ansvarig/-a lärare (ev. telefonnummer)	Farhang Nemati Tel: 019-303095 Mobil: 0702533418
Totalt antal poäng	40
Betyg (ev. ECTS)	The total points are 40. There are 9 questions. At least 24 points are required for grade 3 (pass), 30 points for grade 4 and 35 points for grade 5.
Tentamensresultat	The results will be notified in Studentforum within 15 working days after the exam.
Övrigt	You can write your answers in English or Swedish.

Good Luck!

Questions

1. (3 points)

Consider the class MyClass:

```
class MyClass
{
private:
  int r1;
  int getR1(){ return r1; }
protected:
  int t1;
public:
  int b1;
  int getB1() { return b1; }
  int getT1() { return t1; }
};
```

In the following main() which statements are ok and which one are error. Select OK or Error accordingly. Each statement is identified by the letter in the comment following the statement.

```
int main()
{
  MyClass mc;
  mc.r1 = 2;//a
  std::cout << mc.getR1();//b
  std::cout << mc.getB1();//c
  mc.t1 = 2;//d
  mc.b1 = 2;//e
  std::cout << mc.getT1();//f
};
```

	OK	Error

		OK	Error
A	a	○	○
B	b	○	○
C	c	○	○
D	d	○	○
E	e	○	○
F	f	○	○

2. (4 points)

Define a class (name it *MyST*) with an appropriate constructor and a destructor. It must have two data members as follow:

a. *totalObj*: keeps the total number of objects that have been instantiated from the class since the start of the program.

b. *currentObj*: keeps the number of objects of the class that currently exist. It means it holds the number of objects that have been constructed but not destructed yet.

c. What will be the out put of the following code?

```
using namespace std;

int main()

{

  MyST m1;

  cout << m1.totalObj << " " << m1.currentObj << endl;


  if(true){

    MyST m2;

    cout << m2.totalObj << " " << m2.currentObj << endl;

    MyST m3;

    cout << m3.totalObj << " " << m3.currentObj << endl;

  }

  MyST m4;

  cout << m4.totalObj << " " << m4.currentObj << endl;

};
```

3. (6 points)

Considering the definition of the class MyArray below, it has a few problems; 1) It has memory leak problem, 2) it has out of range access to array problem, and 3) it does not check if an object is constructed with a negative size. Modify the definition of the class to:

a. Remove the memory leak problem.

b. Throw an exception when out of range access to the array happens.

c. Throw an exception if an object is instantiated with a negative size.

```
class MyArray
{
private:
  int* m_array;
  int m_size;
public:
  MyArray(): m_size(0), m_array(nullptr) {}
  MyArray(int size): m_size(size)
  {
    m_array = new int[m_size];
  }
  int& operator[](int index)
  {
     return m_array[index];
  }
};


//Example to show the problems with the current definition.
int someFunction(int i)
{
  MyArray mArr(-10); //It would be a problem

  return mArr[i]; //Out of range access if i < 0 or i >= m_size;
}
//When we return from someFunction, we will have memory leak problem
```

B *I* U ☰ ☰ á

0 / 10000 Word Limit

4. (5 points)

Consider the following class:

```
class Person
{
private:
  std::string m_name;
  int m_age;
  std::string m_address;

public:
  Person(std::string name, int age, std::string address)
    :m_name(name), m_age(age), m_address(address)
  { }
  std::string getName(){ return m_name;}
  int getAge(){ return m_age;}
  std::string getAddress(){ return m_address;}
};
```

a. Overload insertion operator << so that objects of the class can be written to any output stream derived from std::ostream, such that name, age, and address of an object are written to the stream. See the example below.

b. Overload operator + for class Person so that an integer number can be added to an object of Person. The operation will increase the age of the person by the added value. It should return a reference to the person object. See the Example.

Example:

```
using namespace std;
int main()
{
  Person p1("Harry Potter", 19, "Hogwartsgatan 17B, Orebro");
  Person p2("Hermione Granger", 18, "Hogwartsgatan 17C, Orebro");
  Person p3("Ron Weasley", 20, "Hogwartsgatan 17A, Orebro");
  cout << p1 << endl << p2 << endl << p3 << endl;

  int a = 11;
  p1 = p1 + a;
  cout << p1.getAge() << endl;
};
```

Output:

```
Harry Potter; 19; Hogwartsgatan 17B, Orebro

Hermione Granger; 18; "Hogwartsgatan 17C, Orebro

Ron Weasley; 20; Hogwartsgatan 17A, Orebro
```

30

0 / 10000 Word Limit

5. (6 points)

Considering the following classes. Which answer would be the output of the code in the following main()?

```
class Base

{

public:

  virtual std::string who() { return "Base"; }

 std::string foo() { return "Base"; }

};

class D1: public Base

{

public:

  std::string who() { return "D1"; }

 std::string foo() { return "D1"; }

};

class D2: public D1

{

public:

  std::string who() { return "D2"; }

 std::string foo() { return "D2"; }

};

class D3: public D2

{

public:

  std::string who() { return "D3"; }

 std::string foo() { return "D3"; }

};


using namespace std;

int main()

{

  D1 d1;

  D2 d2;

  D3 d3;

  Base b;
  Base* bPtr;
  bPtr = &b;
  std::cout << bPtr->who() << " " << bPtr->foo() << std::endl;//a
  bPtr = &d1;
  std::cout << bPtr->who() << " " << bPtr->foo() << std::endl;//b
  bPtr = &d2;
  std::cout << bPtr->who() << " " << bPtr->foo() << std::endl;//c
  bPtr = &d3;
  std::cout << bPtr->who() << " " << bPtr->foo() << std::endl;//d
  D1* dd = new D2();
  std::cout << dd->who() << " " << dd->foo() << std::endl;//e
  delete dd;
  dd = new D3();
  std::cout << dd->who() << " " << dd->foo() << std::endl;//f
  delete dd;
}
```

A

```
Base Base
Base D1
Base D2
Base D3
D1 D2
D1 D3
```

B

```
Base Base
D1 D1
D2 D2
D3 D3
D1 D1
D1 D1
```

C

```
Base Base
Base Base
Base Base
Base Base
Base D2
Base D3
```

D

Base Base

D1 Base

D2 Base

D3 Base

D2 D1

D3 D1

E

Base Base

Base D1

Base D2

Base D3

Base D2

Base D3

F

Base Base

Base Base

Base Base

Base Base

Base Base

Base Base

6. (6 points)

Assume we have the following three interfaces:

```
class OnlineStoreItem

{

public:

  virtual double getPrice() = 0;

  virtual void setPrice(double price) = 0;

};

class Vehicle

{

public:

  virtual std::string getBrand() = 0;

  virtual int getYearModel() = 0;

};

class IComputer

{

public:

  virtual std::string getBrand() = 0;

  virtual double getProcessorPower() = 0;

  virtual int getRAM() = 0;

};
```

a. Define a class called Car that implements interfaces `Vehicle` and `OnlineStoreItem`.

b. Define a class called Computer that implements interfaces `IComputer` and `OnlineStoreItem`.

Add appropriate data members to your Car and Computer classes wherever needed.

c. Define a global function (name it *printPrice*) that prints out the price of an object passed to it. It should take any object of any class that implements interface `OnlineStoreItem.`

d. In a main() function declare one (only one) vector to which you add pointers to a few objects of both classes Computer and Car. Notice that you have to add the pointers of the objects of both classes to the same vector (not two separate vectors). Then iterate through your vector and print the price of all the objects using your function *printPrice*.

0 / 10000 Word Limit

7. (4 points)

Which answer is the correct answer for the following requests?

a. Define a template function that returns the minimum value of its given two arguments:

```
type min(type t1, type t2)
```

b. Define a specialization of the template function for type Person (the class in question 4). The specialized template function should return the person who is younger.

A

a.

```
template<class T>
T min (T t1, T t2)
{
  if(t1 < t2)
      return t1;
 return t2;
}
```

b.

```
template<class Person>
T min (T t1, T t2)
{
  if(t1.getAge() < t2.getAge())
      return t1;
 return t2;
}
```

B

a.

```
template<class T>
class min (class t1, class t2)
{
  if(t1 < t2)
      return t1;
 return t2;
}
```

b.

```
template<class Person>
class min (class t1, class t2)
{
  if(t1.getAge() < t2.getAge())
      return t1;
 return t2;
}
```

C

a.

```
template<class T>
T min (T t1, T t2)
{
 if(t1 < t2)
    return t1;
 return t2;
}
```

b.

```
template<>
Person min (Person t1, Person t2)
{
 if(t1.getAge() < t2.getAge())
    return t1;
 return t2;
}
```

D

a.

```
template<int T>
T min (T t1, T t2)
{
 if(t1 < t2)
    return t1;
 return t2;
}
```

b.

```
template<Person>
Person min (Person t1, Person t2)
{
 if(t1.getAge() < t2.getAge())
    return t1;
 return t2;
}
```

E

a.

```
template<>
T min (T t1, T t2)
{
  if(t1 < t2)
      return t1;
 return t2;
}
```

b.

```
Person min (Person t1, Person t2)
{
  if(t1.getAge() < t2.getAge())
      return t1;
 return t2;
}
```

8. (3 points)

In the following code which design pattern is applied?

```
class Server
{
public:
  void serve(const IRequest& cl)
  {
    cl.execute();
  }
  //...
};

class IRequest
{
public:
   virtual void execute() const = 0;
};

class Client1: public IRequest
{
public:
  //...

  void execute() const
  {
    std::cout << "Client1" << std::endl;
  }
  //...
};

class Client2: public IRequest
{
public:
  //...

  void execute() const
  {
    std::cout << "Client2" << std::endl;
  }
  //...
};

int main()
{
  //...
  Server srv;

  srv(Client1());
  srv(Client2());
  //...
```

```
 return 0;
};
```

A Observer

B Command

C Singleton

D Adapter

E Factory Method

9. (3 points)

Which answer is a correct about the Single Responsibility Principle (SRP)?

A If S is a subtype of T, objects of S can substitute objects of T in a program without altering the properties of the program.

B A module (class) should have one, and only one, reason to change.

C Split very large interfaces into multiple smaller and more specific interfaces.

D Software entities should be open for extension but closed for modification.